AFRL-RX-WP-TR-2012-0347

CLOSED LOOP ANALYSIS META-LANGUAGE PROGRAM (CLAMP)

Mangus Chisterson

Intentional Software Corporation

MAY 2012
Final Report

AIR FORCE RESEARCH LABORATORY
MATERIALS AND MANUFACTURING DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7750
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

Qualified requestors may obtain copies of this report from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RX-WP-TR-2012-0347 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

//SIGNED//	**//SIGNED//**
C. BRANDON LOVETT, Program Manager	SCOTT M. PEARL, Branch Chief
AFRL/RXMS	AFRL/RXMS

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
May 2012	Final	19 May 2011 – 31 May 2012

4. TITLE AND SUBTITLE
CLOSED LOOP ANALYSIS META-LANGUAGE PROGRAM (CLAMP)

5a. CONTRACT NUMBER
FA8650-11-C-7129

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
62303E

6. AUTHOR(S)
Mangus Chisterson

5d. PROJECT NUMBER
3000

5e. TASK NUMBER
00

5f. WORK UNIT NUMBER
M0129700

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Intentional Software Corporation
2821 Northup Way, Suite 250
Bellevue, WA 98004-1439

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory
Materials and Manufacturing Directorate
Wright-Patterson Air Force Base, OH 45433-7750
Air Force Materiel Command
United States Air Force

Defense Advanced Research Projects Agency/ (DARPA)
3701 N. Fairfax Drive
Arlington, VA 22203-1714

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
AFRL/RXMS

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)
AFRL-RX-WP-TR-2012-0347

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES
The U.S. Government is joint author of this work and has the right to use, modify, reproduce, release, perform, display, or disclose the work. PA Case Number and clearance date: 88ABW-2012-3762, 5 July 2012. This document contains color.

14. ABSTRACT
The Closed Loop Analysis Meta-language Program (CLAMP) sought to develop an Adaptive Vehicle Make (AVM) Workbench, including: 1) A set of common foundation meta-languages that describes a subset of relevant META Design languages and iFAB Foundry languages and relationships between them. 2) A meta-language analyzer that processes models in those languages and provides closed loop manufacturability feedback to designers. 3) Synthesized manufacturability constraints with a set of foundry libraries. The constraints (and design rules) are expressed independently of the chosen design language(s).

15. SUBJECT TERMS
Meta-language, design, foundry, manufacturability

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT	b. ABSTRACT	c. THIS PAGE			C. Brandon Lovett
Unclassified	Unclassified	Unclassified	SAR	34	19b. TELEPHONE NUMBER *(Include Area Code)* N/A

Table of Contents

List of Figures

Summary

The Closed Loop Analysis Meta-language Program (CLAMP) sought to develop an Adaptive Vehicle Make (AVM) Workbench, including:

1) A set of common foundation meta-languages[1] that describes a subset of relevant META Design languages and iFAB Foundry languages and relationships between them.
2) A **meta-language analyzer** that processes models in those languages and provides closed loop manufacturability feedback to designers.
3) **Synthesized manufacturability constraints** with a set of foundry libraries. The constraints (and design rules) are expressed independently of the chosen design language(s).

The AVM Workbench is built on technology developed by Intentional outside of this DARPA program that allows an unlimited set of **integrated design and foundry languages**. It uses consistent **cross language representation** and allows **substitution of different Design and Foundry languages**.

The picture below shows an overview of the languages developed and integrated. The main focus has been on:

- **Product Language** – Language to describe a product model as output from a design process using one of the META tool chains and input to an iFAB Foundry. This product model is used to automate the manufacturability feedback to META designers from one or more iFAB Foundries.
- **Manufacturing Process Language** – Language to describe a synthesized version of the iFAB/C2M2L Manufacturing Model Libraries (MML) which is used to compute manufacturability feedback.
- **Foundry Language –** Language to describe a Foundry and its capabilities expressed in terms of resources and what manufacturing processes it can perform.
- **Process Plan Language –** Language to describe instantiated process plans of how to manufacture a certain product in a specific foundry.

The languages that are used as integration points into these foundation languages are also illustrated in Figure 1, e.g. Modelica, SysML, CyPhy, M-SysML, etc. These are existing languages or languages being developed under the DARPA AVM program that we have built integration points to.

[1] We define a meta-language as a language that operates on other languages.

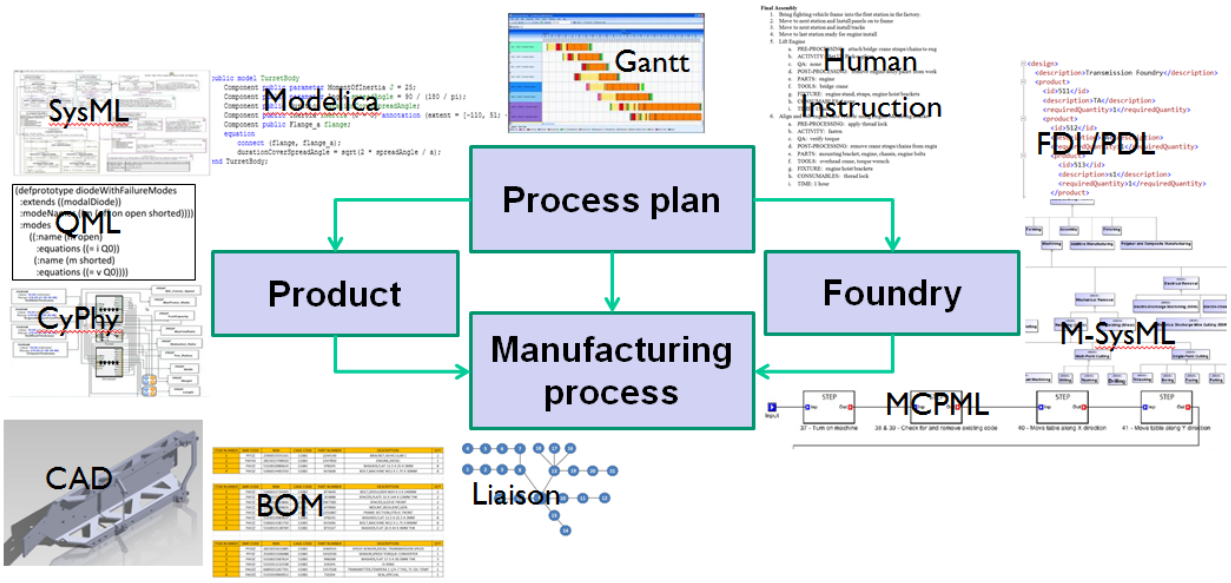


Figure 1. Overview of Foundation languages developed. The arrows shows dependencies between the various foundation languages.

The foundation languages are used to calculate manufacturability feedback at various level of fidelity for a designer to assess their submitted design. At the highest level we want to be able to compute answers like:

Transit time $= \boldsymbol{min}\, \boldsymbol{T_{Stop} - T_{Start}}$

Foundry Cost $= \boldsymbol{min} \sum_{k=1}^{n} \boldsymbol{P_{M_k}}$

Unit Cost $= \boldsymbol{min} \sum_{k=1}^{n} \boldsymbol{P_{R_k}}$;

$$\boldsymbol{R = \{COTS,\ materials,\ consumables\}}$$

Introduction

Using Intentional's Language Workbench, Intentional Domain Workbench (IDW), we researched and developed an AVM Workbench that supports iFAB Technical Area 3 objectives. We collaborated with other META and iFAB teams to implement (a subset of) their languages in a substitutable way, thereby allowing these languages to be integrated, focused, adaptable, and more plentiful. The research showed that the old way of processing language models through information interchange formats and transformation tools, which create data redundancy, can be a thing of the past. Instead we give designers and foundry engineers an integrated experience of a rapid, semi-autonomous closed loop feedback process of manufacturing constraints to a design. The AVM Workbench implements language schemas for the META languages and iFAB languages. All language schemas use a small set of foundation languages to express common concepts like math and physical properties. An Analysis meta-language is used to express manufacturability constraints and design rules that apply across models expressed in any of the design or foundry languages.

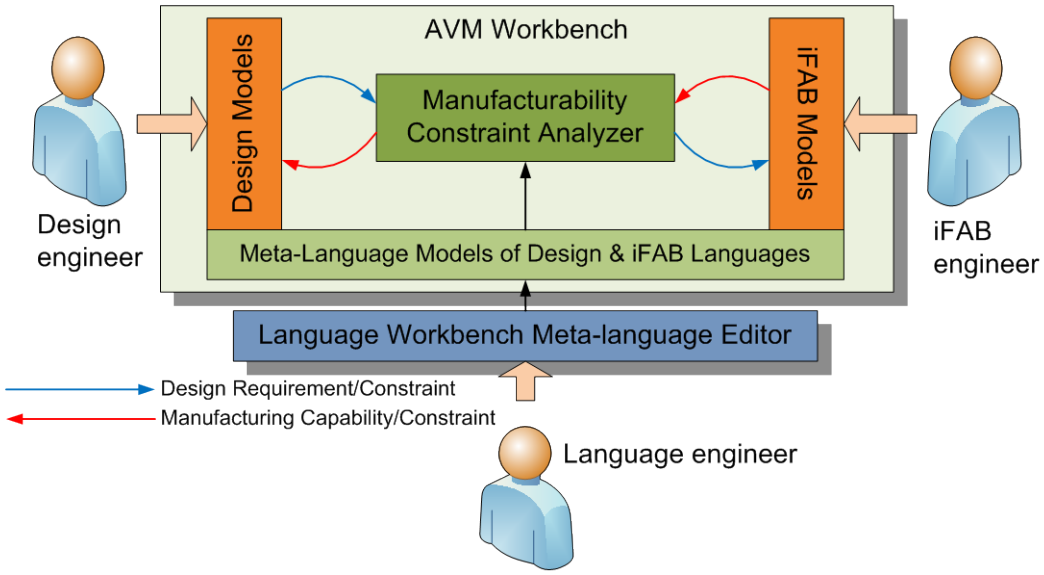


Figure 2. AVM Workbench. Design and iFAB models integrated through a meta-language. Manufacturability Constraint Analyzer runs off the meta-language models.

CLAMP and the AVM Workbench illustrated a novel, scalable approach for design and manufacturing engineers using independent, yet inter-dependent, languages to optimize vehicle design and foundry configuration. In fully developed and deployed CLAMP workflows:

- Design engineers **rapidly respond to manufacturability constraints** thanks to the semi-automatic feedback provided by the AVM Workbench.
- Manufacturing engineers **perform cost analysis earlier in the product lifecycle** – starting from the earliest stages of design.

- Language engineers **improve** existing **relationships** between design and foundry languages – and add new ones – **independent of models** expressed in those languages. (These improvements will result in a continuous strengthening of the relationship between design and foundry over time.)

Rather than relying on manual feedback processes using standalone, general purpose languages and tools, engineers rely on AVM Workbench to express their intent precisely and to respond quickly to the precisely expressed intent of others. Rapid feedback provided by AVM Workbench results in **shorter design iterations** and **well-informed engineers**, both of which lead to better vehicle designs – **vehicles that can be manufactured in small quantities at lower cost with production-level quality.**

Methods, Assumptions, and Procedures

First we performed an analysis of all languages that have been used or proposed from other META and iFAB teams. The following languages represent the language choices for various teams. The result for the META language analysis is summarized in the table below.

META Team	Requirements	Structure/Architecture	Dynamics/Behavior	CAD/Physical
Vanderbilt	DOORS, SysML	CyPhy(CLNG, AML), SysML (SysArch), Formula	Bond Graph, Modelica, Simulink, Matlab	Pro/E
IBM/UTRC	(SysML)	SysML (Rhapsody), MoCC	Excel, OCL, CPlex	?
BAE	(SysML)	SysML (MagicDraw), AMIL	Modelica, Matlab, Simulink	Pro/E
Rockwell Collins		SysML (Sparx), AADL, Lute	EDICT	?
Adventium	(SysML)	SysML (Topcased), AADL	Excel, Modelica	Pro/E

The analysis shows that there is some consensus and convergence, but also a wide variety:

- For structural/architectural modeling, SysML was mentioned by all teams, but each team seems to have chosen a different tool implementation for SysML. SysML is central to some teams, and more peripheral to other teams.
- For dynamic/behavioral modeling, Modelica is mentioned by three of the teams. Excel, Matlab, Simulink by two teams.
- For CAD, ProE is mentioned by 3 teams. 2 teams made no reference to CAD. Surprisingly, CAD does not seem to be central for any of the teams; they are very peripheral to all teams that mentioned them.
- Requirements were mentioned by a few teams, but no team put a central focus on this. A bit surprising since Requirements Engineering has evolved recently as its own discipline with the recognition that many design problems can be surfaced at the requirements level. SysML is mentioned by a few teams, but SysML does not have deep support for Requirements.
- We did not analyze verification languages as they are outside of the current scope of CLAMP.

We ended up focusing on the following languages: CyPhy, Modelica and SysML. For CAD we ended up focusing on FreeCAD, OpenCascade and SolidWorks. Integrating CAD elements into the meta data of a design is particularly not well developed due to the limitations of current tools and standards (like STEP). Appendix 1 discusses an approach to this discussion that eas explored in our research.

Our focus has been on not only on representing the languages, but also to integrate them. With respect to **integrating SysML and Modelica** language models there have been a number of efforts in the last few years. The following two are the most important ones:

- One effort, ModelicaML[2], is focused on expressing Modelica models as a SysML/UML profile. The benefits would be to allow Modelica models be expressed as graphical models, but the drawbacks are that they require the use of a new SysML/UML profile and therefore would not work with native SysML models.
- Another effort started in December 2008, and is currently ongoing within OMG, is to create an official SysML – Modelica Transformation specification[3]. It is currently under development, and a beta 1 version is currently available for review. This specification also requires additional new profiles to be learned by users. Furthermore, it relies on manual and hand coded transformation steps to be feasible.

Our approach within CLAMP for language integration in general, and for SysML and Modelica integration in particular, is quite different from these efforts in that we try to preserve the intention from the designer in what they have already done by using different languages to express and refine those design intentions. We try to avoid defining new language constructs for designers to learn as part of this process.

For **iFAB languages** we have also done a language analysis. A summary of the result is in the table below.

Team	Assumed input	Factory languages
U of Delaware	CAD: CATIA	LIMS (Liquid Injection Molding Software), SIMULIA, DELMIA, ABAQUS, Labview, Matlab
CMU/UMD/UMI/LM/P&M	CAD?	Human operator instructions, …
Boeing/GM		MCPML (GME based)
Penn State	CAD Trade Space Visualizer	Foundry Description Language (FDL) Product Description Language (PDL)
PARC	CAD (OpenCascade)	Graph Grammar Rules
Georgia Tech	CAD	M-SysML (MagicDraw)

[2] Towards Unified Systems Modeling with the ModelicaML UML Profile. Pop, A., and Akhvlediani, D., and Fritzson, P. International Workshop on Equation-Based Object-Oriented Languages and Tools. Berlin, Germany, Linköping University Electronic Press, 2007
Towards Unified System Modeling and Simulation with ModelicaML: Modeling of Executable Behavior Using Graphical Notations. Wladimir Schamai, Peter Fritzson, Chris Paredis and Adrian Pop, Modelica Conference 2009.

[3] An Overview of the SysML-Modelica Transformation Specification. Paredis, C.J.J., Bernard, Y., Burkhart, R.M., de Koning, H.-P., Friedenthal, S., Fritzson, P., Rouquette, N.F., and Schamai, W.in Proceedings of the 2010 INCOSE International Symposium, Chicago, IL, July 12-15, 2010.
SysML-Modelica Transformation, FTF Beta 1. OMG document ptc/2010-11-30

Results and Discussion

iFAB Library integration

Our focus has been on integrating the iFAB Libraries from Boeing and GTech. The import of data from Boeing and GA Tech follows a common process with three main steps. First, the schema information for the library is imported. Second, this imported schema is used to guide the import of instance data. In the MLibrary case, the imported schema is described in the IDW Schema language, and this guides the import of a raw JSON dump from the database instance data. The schema may come from either a MagicDraw file or the database representation of the schema;, each has its own conversion. In the MCPML case, the imported schema is described in the language of DBML, and is used to both guide the queries to the database and the form the results of the queries take once imported.
Finally, the third step in either case, once all the instance and schema data is represented in the system, is to transform the entire document concurrently to our foundation languages for manufacturing process and products, which contain representations of both schema and instance information. For example, operationtypes and attributedefs come from the MSysML schema information, while resourcetypes come from the machine instance data in both libraries

Boeing: We have provided feedback on the Boeing effort on their Assembly library, see below. Below are some screenshot on the Bopeing library data and how it is integrated into the iFAB foundation languages together with GTech.

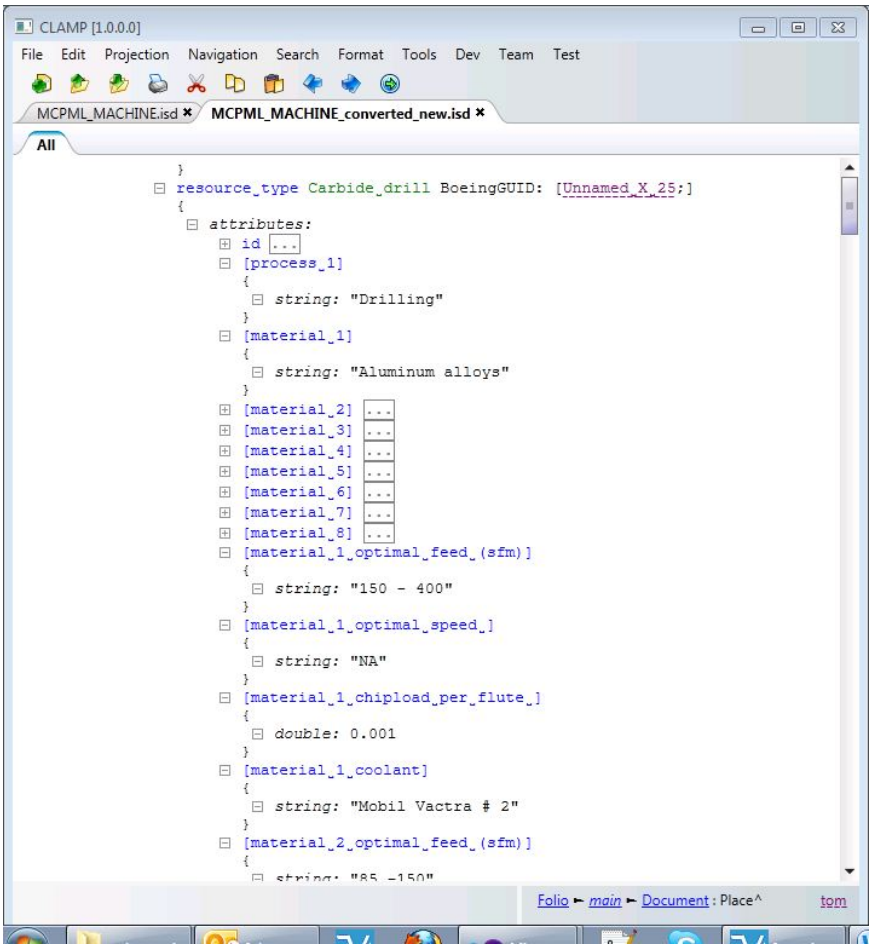


Figure 3. Boeing MCPML library Source Data

Boeing contributed three files since the January PI meeting:

1) 2012-01-26_MCPML_Queries.docx
2) 2012-02-01_iFAB_MCPML_Interface_d09.docx
3) Human Assembly Process Model_11.xlsx (2012-01-26)

Also discussed is the "Human Capabilities for Jan PI meeting.pptx" presented at the January PI meeting, hereafter abbreviated "Human Capabilities".

1) The queries in the first file are reasonable in content and spirit. General comments:

- Most "move" operations are overspecified and should be considered optional. Emphasize other processes. Moves can usually be discovered by rules.
- If transport is not considered, the "foundry x" clause is not necessary, since it amounts to possessing the resources and consumables in question.
- All cost and time numbers should be limited to one significant figure.
- All references to "configure" and "configuration" (e.g. of resources) should be considered optional, since they are hard to interpret and less useful than some of the other items.

What is meant by a feature in the queries? If possible, Boeing should address feature nature not as a standalone contribution but in the terms that Georgia Tech has begun to define for describing features. A plan for integration with GATech features is forthcoming. Illustrative examples of (product,feature) pairs are essential for all questions that relate features.
Possible/not possible questions are more useful now than "how long". How long questions' precision should probably be limited to one significant figure, based on the current limits of discriminating between the different operations in relation to the product. A list of the questions classified as Important / Optional is appended below.
2) Boeing's modeling objectives will be adequately validated with a simple batch programs. Development of a web service should not be necessary. Integration efforts are straightforward starting from a model data set and an interpreting program. Based on existing efforts, configuration/model data for the programs may be most easily sourced from spreadsheets. Use a well crafted attribute-driven access mechanism for spreadsheet data retrieval. For example, looking up row 9 of the Taxonomy table could be achieved by a helper function:

```
SpreadsheetGet(
        "Activity","Adust",
        "Object Category", "PPE"
        "Plant Equipment, Material and Tool", "Ring Cover")
```

This would return a dictionary from which a "Hand Posture" of "Precision Grip" could be obtained. Programs written against spreadsheets should have a routine to check for errors and report when the spreadsheet is not well formed.
The top two considerations for choosing contribution formats of Spreadsheets, XML, MCPML, or something else should be the (anthropometry or other) experts' productivity in:

1) crafting data
2) applying crafted data to configure decision support algorithms.

Collecting feedback early and often is essential here.
3) Human assembly process model should focus on getting a way to programmatically evaluate the various anthropometric information. The biggest easiest improvements fall into two categories.

3.a) Relate the lower fidelity postures to a higher fidelity model, for purposes of documentation and maintaining valid correspondence between the low and high fidelity models.

For example, the only information on Riveting is:

Rivet; Tool; Rivet Gun; Obtain, Align, Activate; Maximum Grip; move; ; get; ; move; ; put; ; move; ; put; ; load

How is "Maximum grip" different from "Full Hand Power Grip"?

The January "Human Capabilities" presentation referred to the high fidelity model SANTOS, yet there is no associativity with the low fidelity model summarized in the spreadsheet. The low fidelity model will not be useful without associativity. One way to achieve associativity would be to encode coordinates of SANTOS in the spreadsheet. The 25 degrees of freedom of Santos' hand could be recorded as 25 named columns in a spreadsheet. Each row would encode one hand gesture and give the 25 real numbers for describing the gesture. A script should be written to make a thumbnail of each hand posture for documentation purposes.

Building on SANTOS association, strength and fatigue thresholds for a low fidelity model should be partially (and in some cases wholly) extractable from SANTOS by programmatic means.
Instead of specifying explicit sequences, consider either raising the level of abstraction by providing precedence (aka dependency) relationships, or by focusing on the aspects of operations other than their final sequence.
3.b) Focus on decision thresholds for matching manufacturing processes to annotated products. The "Reach Zone Abstractions" (slide 7) from "Human Capabilities" seem to be a reasonable starting point. The degrees of freedom of each threshold have to provide the ability to configure a program to match or not match in a given product context. Concretely describe some artificially simple products and/or draw in CAD so that particular grips or reaches can be matched or not matched based on programs relating those degrees of freedom. For example:
"Grip type 123" means the ability to:

- maintain grasp an object up to 1"
- move inside a cylinder of 4.5" radius and 6" depth
- rotate the object (e.g. a nut) for threading inside the hole
- torque to 1 lb-in
- repeat operation at least 20 times per hour and 100 times per shift

Checking such a description against a product can now be decomposed by a program which can identify one or more cylinders against a product feature, and a rule to check the attributes above on the cylinder. The cylinder is not the only kind of geometry appropriate useful to this kind of model. Intersections of 2-3 perpendicular half planes would probably also be a useful starting point.
A grasp fitting inside a cylinder is an example of a bounding constraint, whereas slide 8 of the January "Human Capabilities" is a reaching constraint. These two kinds of constraints and others are all useful for modeling process matching.
Again, favor precedence relationships over explicit sequences, but both are optional.
</Boeing feedback>
We are continuing to work closely with Boeing.

GTech. We have received an updated MongoDB and models from GTech and are working on the integration. So far the M-SysML schema is used to guide the import of the instance data from the GA Tech M-Library. In order for this to happen, the workbench converts the schema from a generic XML-format representation (M-SysML_v66_2012-01-18.uml, obtained from GA Tech group as an export from MagicDraw) to the workbench's internal schema language. This conversion makes use of a general schema for SysML, as well as a custom transformation from SysML to the schema language. This internal representation of the M-SysML domain can then be processed by the workbench to generate code for the import of a raw JSON-format dump of the M-Library into this M-SysML domain. In this internal representation, additional metadata can be attached to the instance data based on the schema, including the units of the attributes, or attributes that are expected but missing from the instance data. Next steps was to transform

schemas and data into CLAMP foundation languages and thus reconcile with Boeing iFAB models.

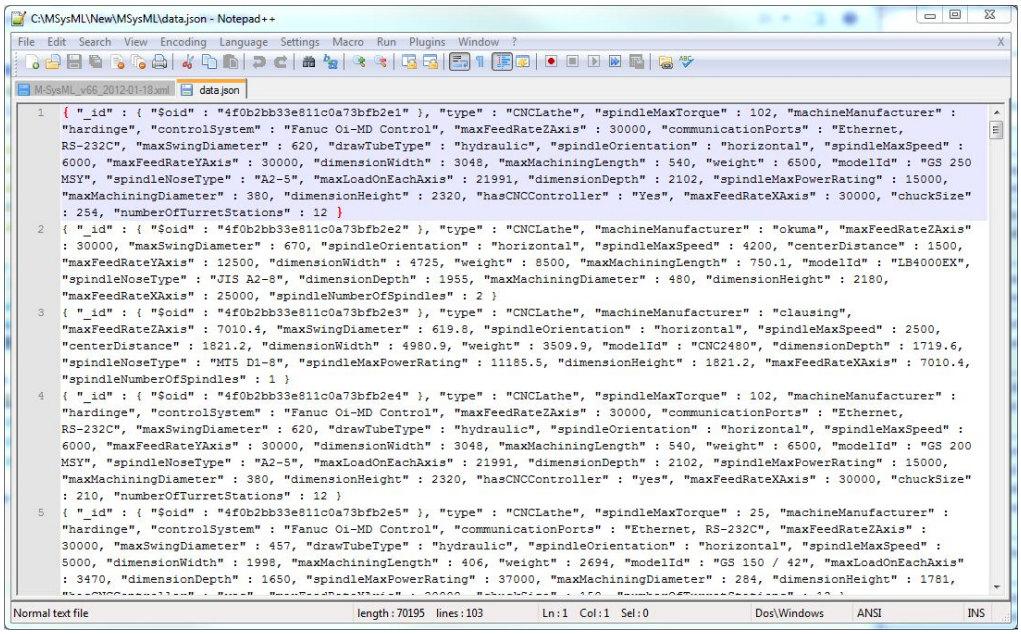


Figure 4. M-SysML raw MongoDB data exported as json objects.

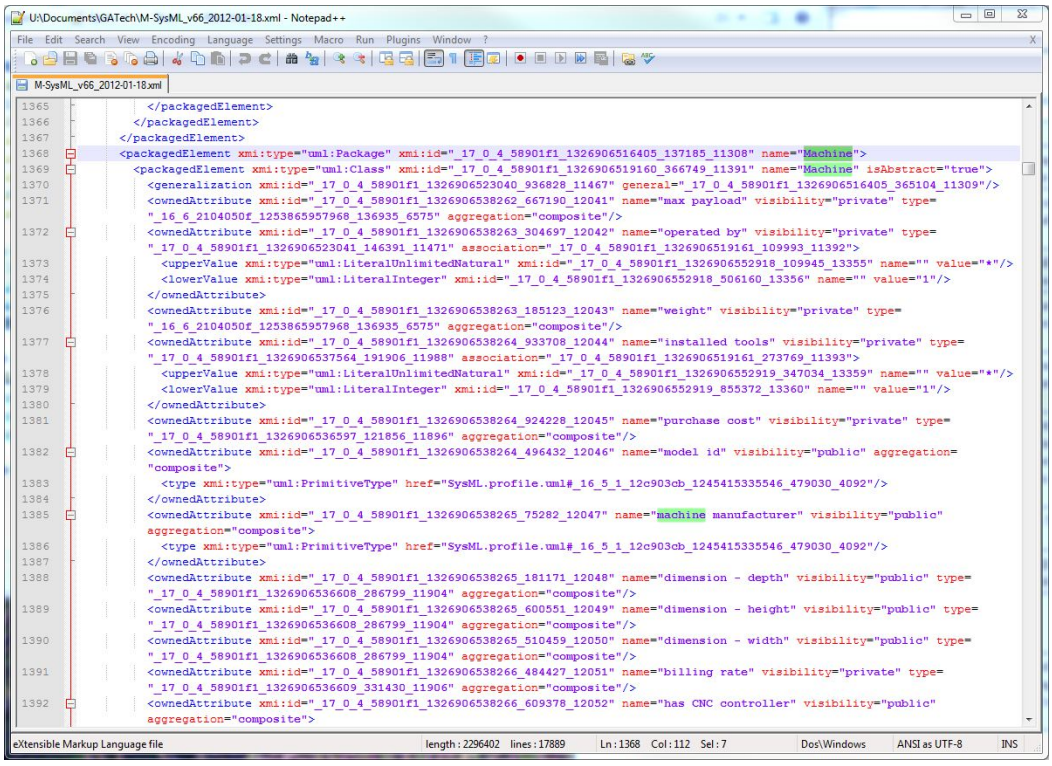


Figure 5. M-SysML uml schema that should be used to interpret the data above.

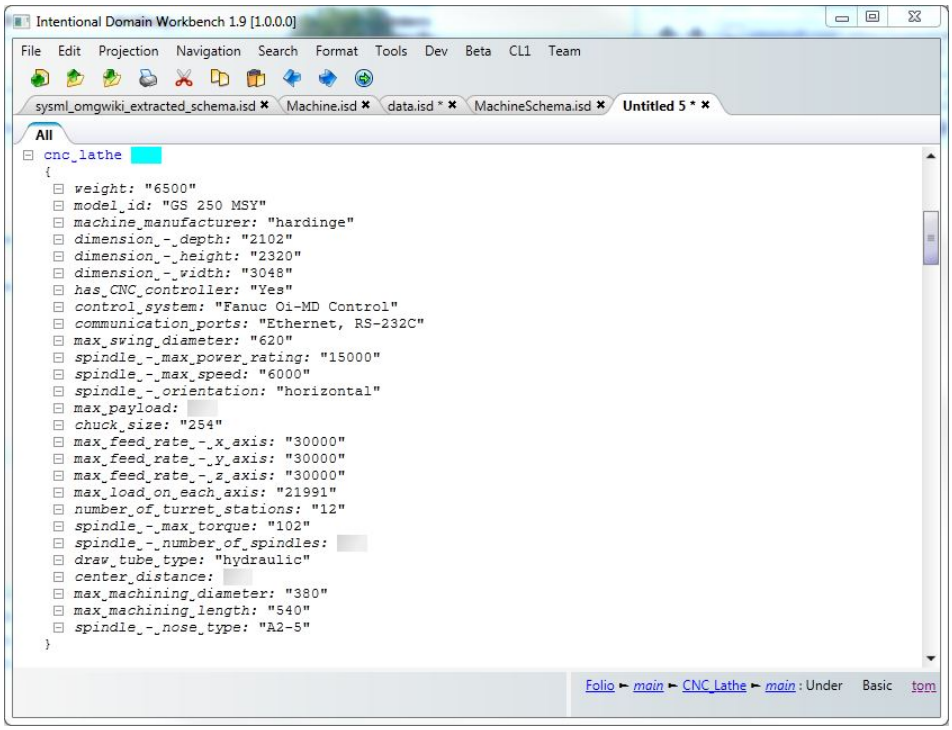


Figure 6. Some example data imported into the AVM Workbench.

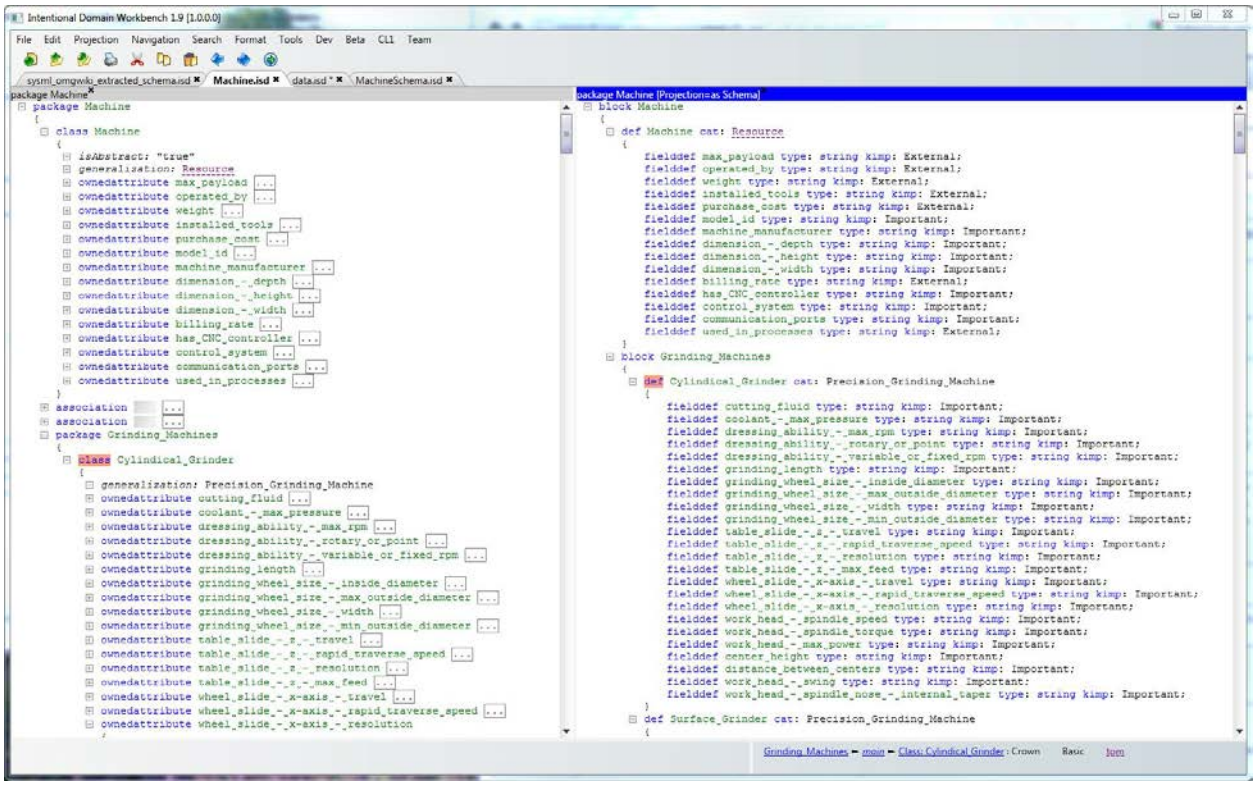


Figure 7. The library data as imported and transformed into the normalized Foundry language (note that the projection has not been tweaked to a desirable syntax).

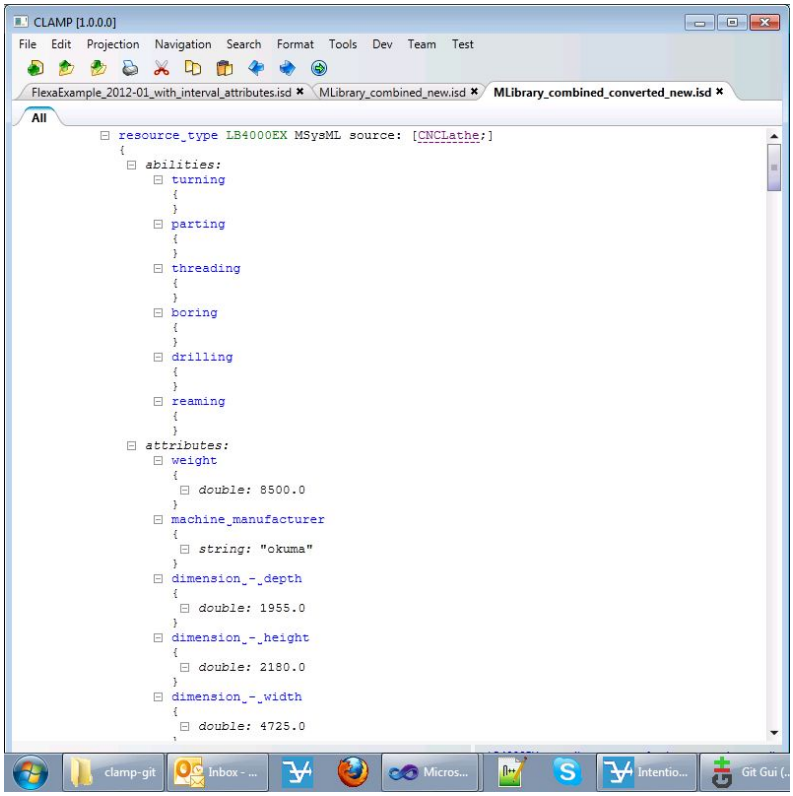


Figure 8. GTech MSysML machine instance data in the foundation languages.

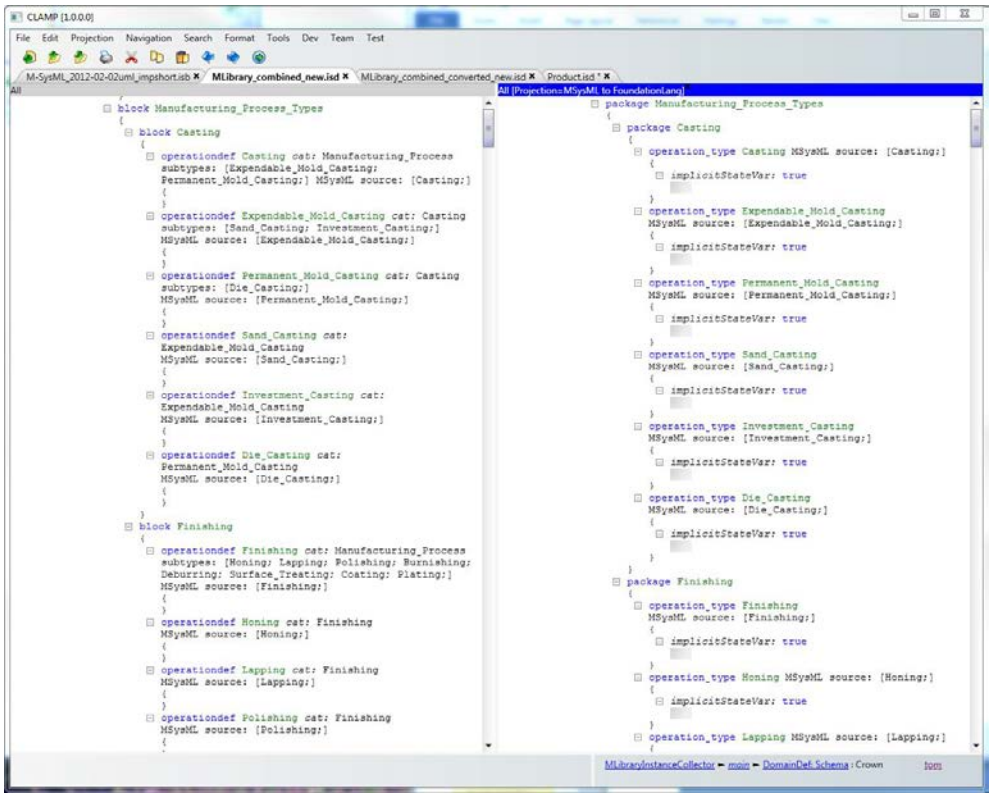


Figure 9. GTech Processes in the Manufacturing process language.

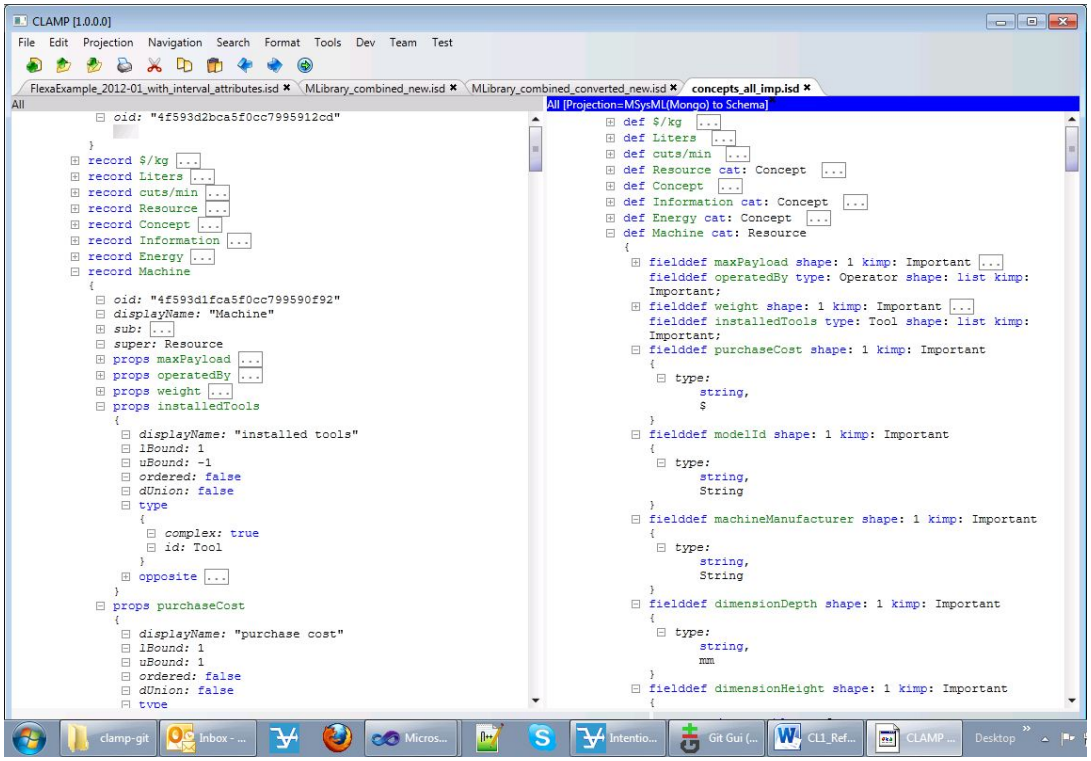


Figure 10. MSysML-Mongo to Foundation language schema mapping.

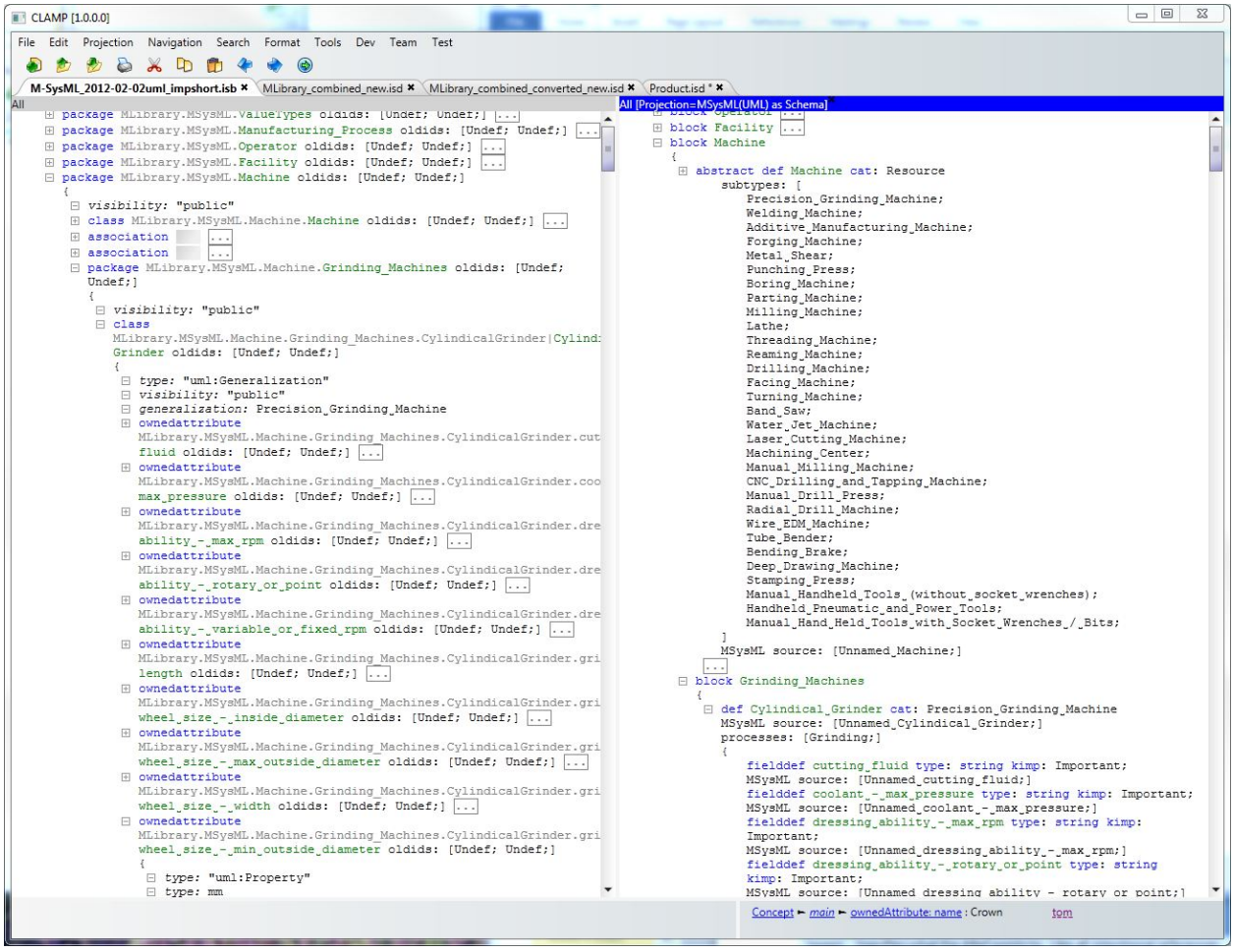


Figure 11. MSysML-UML schema to foundation language schema.

Foundry Configuration Exercise

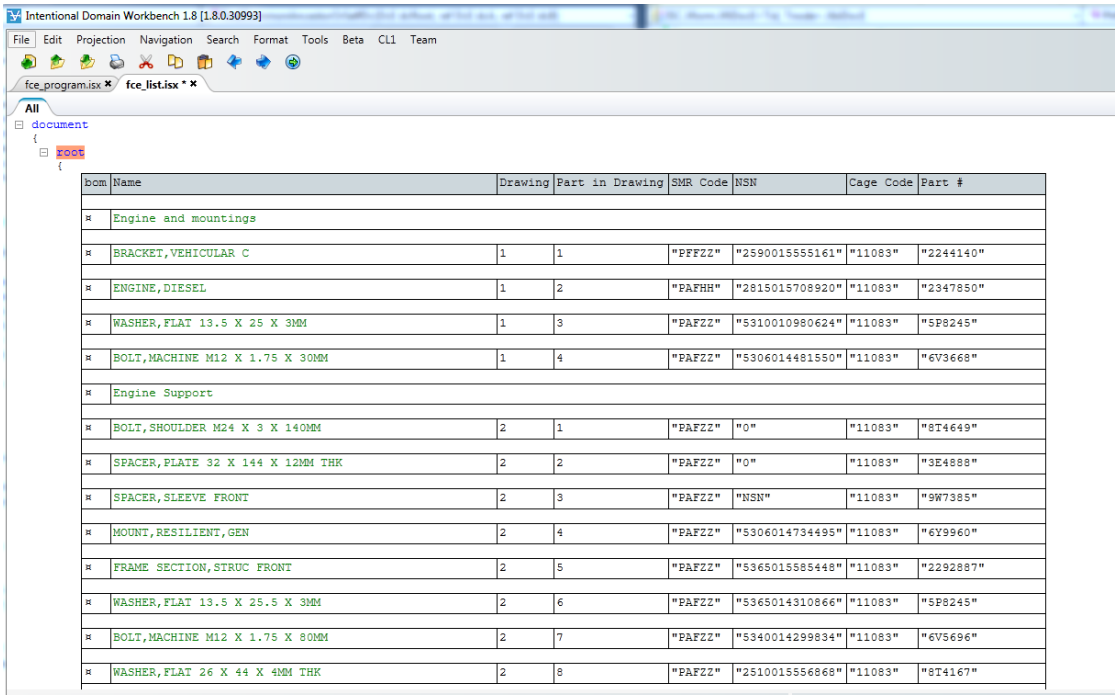
The first test in the program was to apply the various iFAB Languages to a Foundry Configuration Exercise. The Foundry Configuration Exercise (FCE) began as communication by email and phone. The scope was set to start with process planning with the tangible example to assemble the engine and transmission of a Caterpillar 966 front loader. The results of the communication grew to include some written artifacts in PDF, Excel and Powerpoint.

Our main contribution to this conversation was to collect the interests of the various participants and aggregate some of the best sources of data into one machine processable corpus that would be useful to all. This data is in the form of xml. It contains data harvested from the Cat 966 field manual by Jonathan Vance's team at Boeing, Powerpoint slides also from that team, and an M4 bill of materials contributed by Penn State.

We are seeing our role in the continuing exercise as bridging the gap between the various participants. Part of the bridging exercise was spending several hours in calls and email (and in one case an onsite visit) trying to assess the data needs of the various participants within the exercise scope. While we provided an xml file, we also provided a translation so that if extra

attributes were added to the original spreadsheet data, that it could flow through to the generated xml, implanted in Excel using worksheet functions. So the other part of the bridging exercise was in some sense teaching the craft of structured data creation and its benefits, by example. The response so far has been positive.

We plan on improving the xml workbook for the scope of the current exercise but not longer. The reason is that we believe we can do better in the longer term using our language workbench technology. In a purely technological sense, it would have been better for all participants to download our system and use it for data curation in the FCE. That is one of our system's many purposes. However, we wanted to lower the learning curve for busy people who often hardly have time to take the 30 minute phone calls. We wanted them to be empowered to own and improve, and recontribute the shared data, so we illustrated the principles of structured editing with Excel worksheet functions. We hope that the xml data and spreadsheet data will help people understand the benefits of structured editing and how they can have an even better experience by incorporating our technology into their work, see Figure below. Note now that the particular parts and their description are now references to their correct element from a component library or META design model.



bom	Name	Drawing	Part in Drawing	SMR Code	NSN	Cage Code	Part #
¤	Engine and mountings						
¤	BRACKET,VEHICULAR C	1	1	"PFFZZ"	"2590015555161"	"11083"	"2244140"
¤	ENGINE,DIESEL	1	2	"PAFHH"	"2815015708920"	"11083"	"2347850"
¤	WASHER,FLAT 13.5 X 25 X 3MM	1	3	"PAFZZ"	"5310010980624"	"11083"	"5P8245"
¤	BOLT,MACHINE M12 X 1.75 X 30MM	1	4	"PAFZZ"	"5306014481550"	"11083"	"6V3668"
¤	Engine Support						
¤	BOLT,SHOULDER M24 X 3 X 140MM	2	1	"PAFZZ"	"0"	"11083"	"8T4649"
¤	SPACER,PLATE 32 X 144 X 12MM THK	2	2	"PAFZZ"	"0"	"11083"	"3E4888"
¤	SPACER,SLEEVE FRONT	2	3	"PAFZZ"	"NSN"	"11083"	"9W7385"
¤	MOUNT,RESILIENT,GEN	2	4	"PAFZZ"	"5306014734495"	"11083"	"6Y9960"
¤	FRAME SECTION,STRUC FRONT	2	5	"PAFZZ"	"5365015585448"	"11083"	"2292887"
¤	WASHER,FLAT 13.5 X 25.5 X 3MM	2	6	"PAFZZ"	"5365014310866"	"11083"	"5P8245"
¤	BOLT,MACHINE M12 X 1.75 X 80MM	2	7	"PAFZZ"	"5340014299834"	"11083"	"6V5696"
¤	WASHER,FLAT 26 X 44 X 4MM THK	2	8	"PAFZZ"	"2510015556868"	"11083"	"8T4167"

Figure 12. BOM natively projected in the AVM Workbench rather than Excel.

Perspectives

CMU

The CMU team has a focus on process planning on a narrow domain. As far as we can tell, the CMU team plans on emitting a process plan with all degrees of freedom removed. Welding contains, as does any specific manufacturing process, certain assumptions about how the work may proceed. They are strongly committed to automating the fabrication of their product.

However, I am not quite sure if they will produce any process planning artifacts usable by other participants. They have been very interested that the FCE data contains all the aspects that they need to consume, but less eager to talk about what others would like to consume. Since the CMU team will be analyzing every nut and bolt in the foundry configuration exercise, and Penn State is expecting a bill of materials to contain only three nodes for engine, transmission and frame, there may be a possible data interchange between the two teams of only three nodes.

Boeing

Boeing has a background in hands on processes, and have done a great job of getting a lot of that data in one place. We are very curious at about what level their MCPML language will turn out to be actionable. The language conflates many primitive ideas about manufacturing process with other ideas about foundries and instructions. We also are concerned that they have outsourced their specific data entry to a subcontractor at Vanderbilt. The subcontracting has insulated them from being aware of just how much labor is being performed to enter even very simple examples in the MCPML language. We fear that the data entry aspect of MCPML may make the language unusable. We would encourage the AVM program to have another participant create a model in the MCPML language and have Boeing process it by machine in some useful way.

Penn State

Penn State has a background in agent-based architecture and the operations management aspect of manufacturing. Instead of building a computer-guided search through the foundry configuration trade space, they plan on showing a smart user interface to a smart user and allowing the user to completely guide the optimization in the foundry configuration space. Their biggest risk is dependence on a high quality cost function to accurately reflect the assignment of different foundry resources to units of work on the product. While there has been some solicitation of parametric cost data from other participants, we do not see how such a cost function can be created without correct identification of the degrees of freedom of said cost function. The processes Penn State wants to cost are very few: positioning, fixing and inspecting. Penn State would like to roll up their bills of materials to approximately 10 high level nodes (engine, transmission, frame, battery, etc) and decide amongst assembly sequences approximately 9! in number. The kind of decision process Penn State wants to answer is to for example compare these two process plans:

1) Move and fix engine to chassis. Move and fix transmission to engine and chassis.
2) Move and fix transmission to chassis. Move and fix engine to transmission and chassis.

Bulk moving is perhaps the easiest element to cost, since the equipment handling capability will be primarily determined by mass. Fine level positioning is much harder to quantify. The number and type of kinematic degrees of freedom of each fastening will be a big cost contributor. It's a lot easier to ensure that planar surfaces on two objects are in contact than it is to ensure that bolt holes line up. Exhaustive lists of the degrees of freedom and informed framing of the costs will be essential to their objectives.

Suggestions

The FCE illustrated why machine processable data is one of the basic components of interaction between various participants. We recommend creation of more machine readable data to be created, shared, and exchanged among teams. We believe that continued exchange of each participant's core data structures, both input and output, will be essential to the continued successful development of the AVM toolchain.

Requiring implementation of challenge problems pose a concern to the direction of the program, in that a problem which is not in the path of the main research direction of a participant may be distracting to their particular research goals. The upside is of course development of a common set of problems for teams to use in communication both verbally and in software. We believe that the AVM program can expand the number of challenge problems from the current level and still have the upside outweigh the downside.

In cases where it does not make sense for multiple teams to both be consuming as input the same set of data because the distraction from central goals would be too high, the AVM program could instead negotiate a data input/output interchange between groups of teams where the members of the group are defined as either producers or consumers or both. For example, Georgia Tech could produce an input file for CMU to consume, or a CMU analysis program could produce a file for the Penn State team's software to consume. The negotiation should result in a schema and illustrative example.

Beyond a language as syntax, the meaning of any language (also known as its semantics) lies in illustrations of how to perform analysis with data expressed in the language. We recommend an all around higher expectation of illustrative example analysis programs on every contributed machine-readable document. Collaboration between teams is especially relevant to confirm that the illustrative examples are sufficient in number and clarity.

META-iFAB Exercise

This exercise is ongoing as of this writing.

Try CLAMP on Commercial design process

A requirement from the original iFAB BAA was to show that the developed approach is also applicable to non-META design processes. Together with Chalmers we are trying CLAMP on commercial industry manufacturing processes. We have used an example from an existing project with Volvo Aero called FLEXA, to optimize a multi-robot cell with four robots, a set of machining stations and a human working in unison for parts manufacturing, see below. The objective is to design a flexible automation cell that can manufacture an array of parts for a turbine structure from a variable product structure.

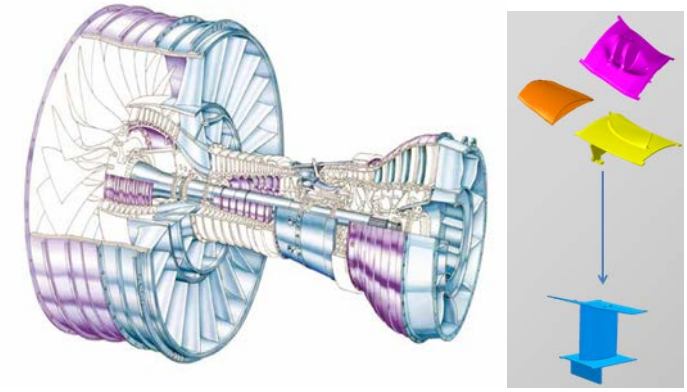


Figure 13. Product to be manufactured.

The Volvo Aero cell layout consists of four robots, one human, a variable number of milling machines, washing machines, deburring machines, measuring stations as well as fixtures, and work tables.

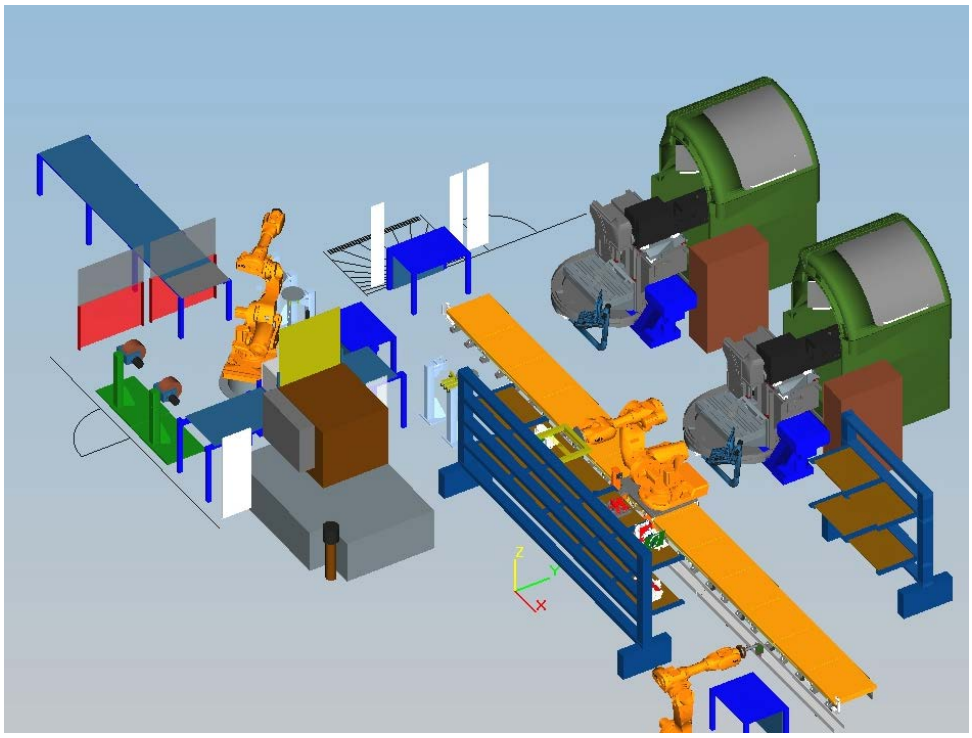


Figure 14. Flexible cell layout.

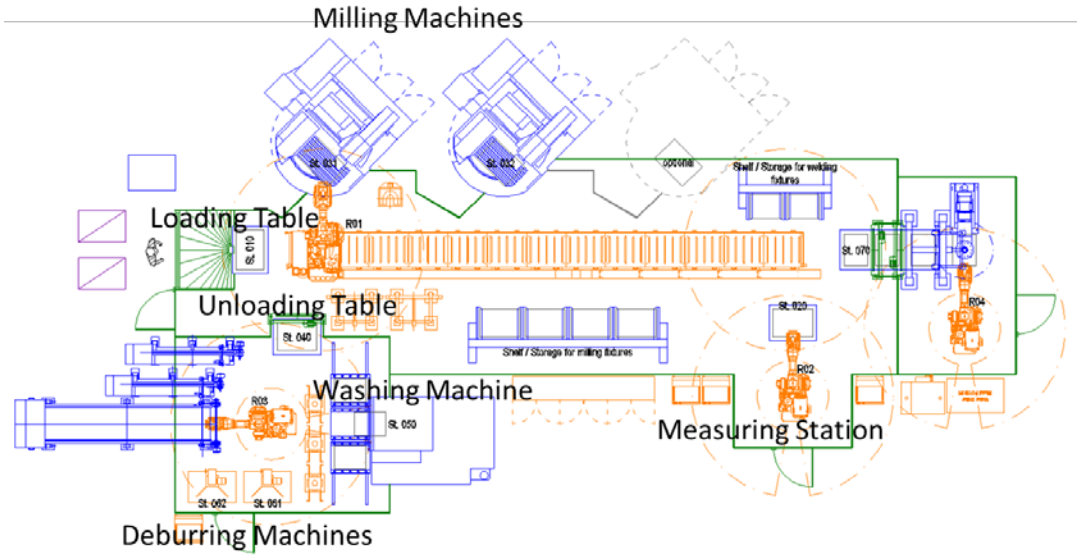


Figure 15. Schematic cell layout.

The operation sequence is:

Process	Sequence of Operation	Process times in minutes	
		Prod. A-C	Prod. D-H
Mtrl handling	Loading of part 1 (hub) into milling fixture	2	2
Parts info	Part identification (reading of serial number)		
Milling	Milling of welding surfaces and locator points	32	32
Milling	Deburring of edges of welding surfaces if required		
Milling	Measuring of machined surface, in milling machine		
Mtrl handling	Unloading of part from milling fixture	2	2
Mtrl handling	Loading of part 2 (shroud) into milling fixture		2
Parts info	Part identification (reading of serial number)		
Milling	Milling of welding surfaces and locator points		42
Milling	Deburring of edges of welding surfaces if required		
Milling	Measuring of machined surface, in milling machine		
Mtrl handling	Unloading of part from milling fixture		2
Mtrl handling	Loading of part 3 (case) into milling fixture	2	2
Parts info	Part identification (reading of serial number)		
Milling	Milling of welding surfaces and locator points	22	22
Milling	Deburring of edges of welding surfaces if required		
Milling	Measuring of machined surface, in milling machine		
Mtrl handling	Unloading of part from milling fixture	2	2
Mtrl handling	No milling on part 4, ready to weld		
Mtrl handling	Loading of part into cleaning station		
Parts info	Part identification (reading of serial number) if required		
Washing	Washing to remove oil/emulsion film. 3 (4) parts	10	10
Brushing	Brushing on hub.	2	2
Brushing	Brushing on case.	2	2
Brushing	Brushing on shroud.		3
Brushing	Brushing on vane.	3	3
Washing	Washing to remove dust after brushing. 3 (4) parts	10	10
Mtrl handling	Loading of parts into welding fixture	8	8
Parts info	Part identification (reading of serial number)		
Welding	Laser welding in Argon atmosphere. 3 (4) parts to 1	20	20
Parts info	Marking of the assembly (new serial number)		
Measuring	Measuring of the assembly	2	2
Mtrl handling	Unloading of assembly from welding fixture	2	2
Measuring	Measuring of the assembly	2	2

This example has been entered into the AVM Workbench and was showed at the latest PI meeting. Below are some screenshots of the manufacturing process library as well as the foundry itself.

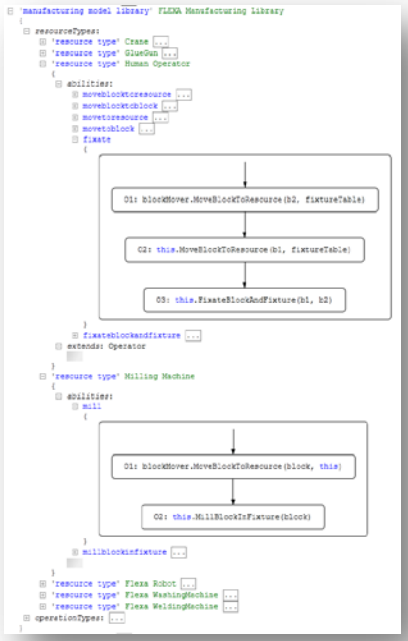


Figure 16. Manufacturing process library for FLEXA cell.

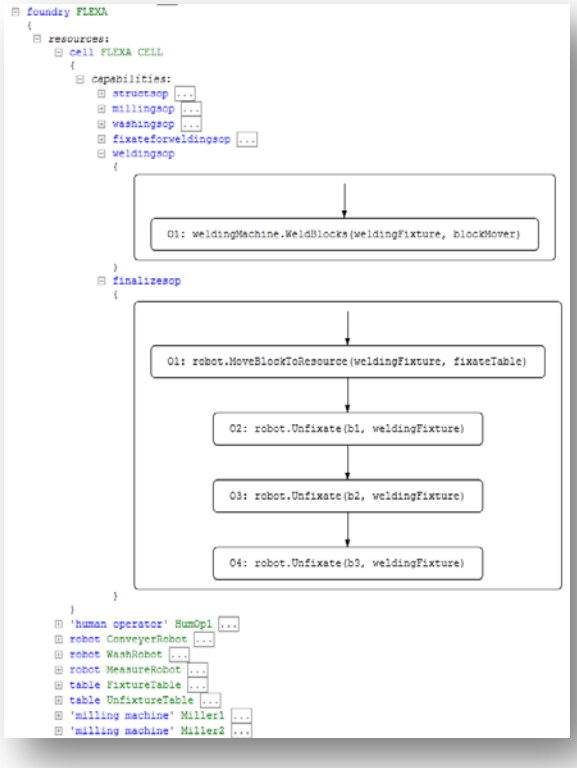


Figure 17. Foundry description for FLEXA cell.

Next is the process plan and some of the primitive operations that will be used for scheduling, control programs and operator instructions.

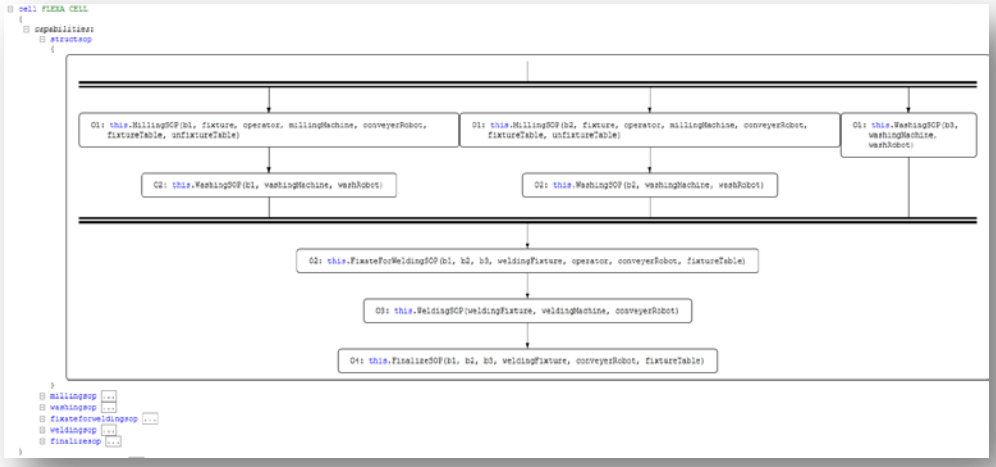


Figure 18. Process plan for FLEXA cell.

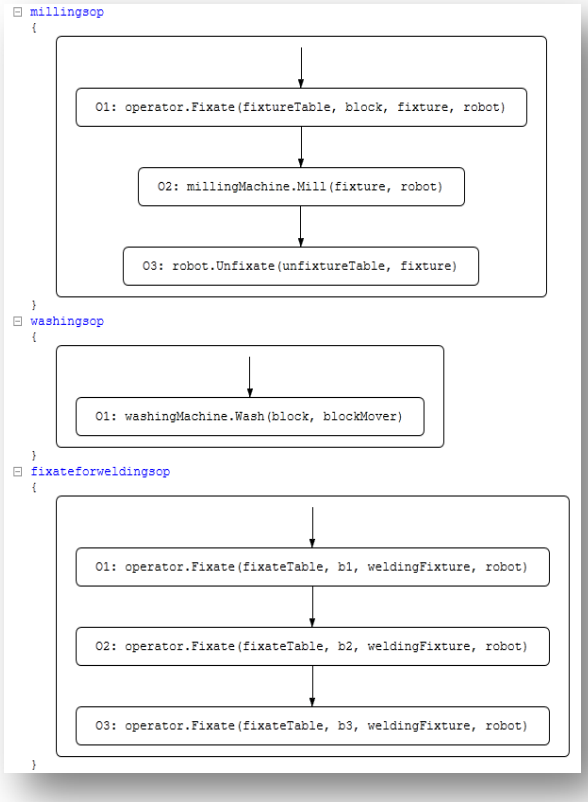


Figure 19. Primitive Manufacturing process operations.

This cell was entered manually into the AVM Workbench and was run and simulated based on the throughput criteria and demoed in Camp Pendleton. Based on that model the best answer was 74 minute cycle time!

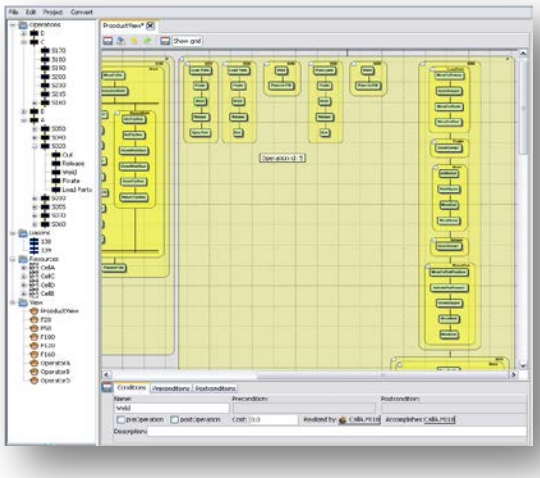


Figure 20. Sequence Planner showing possible parallel operations.

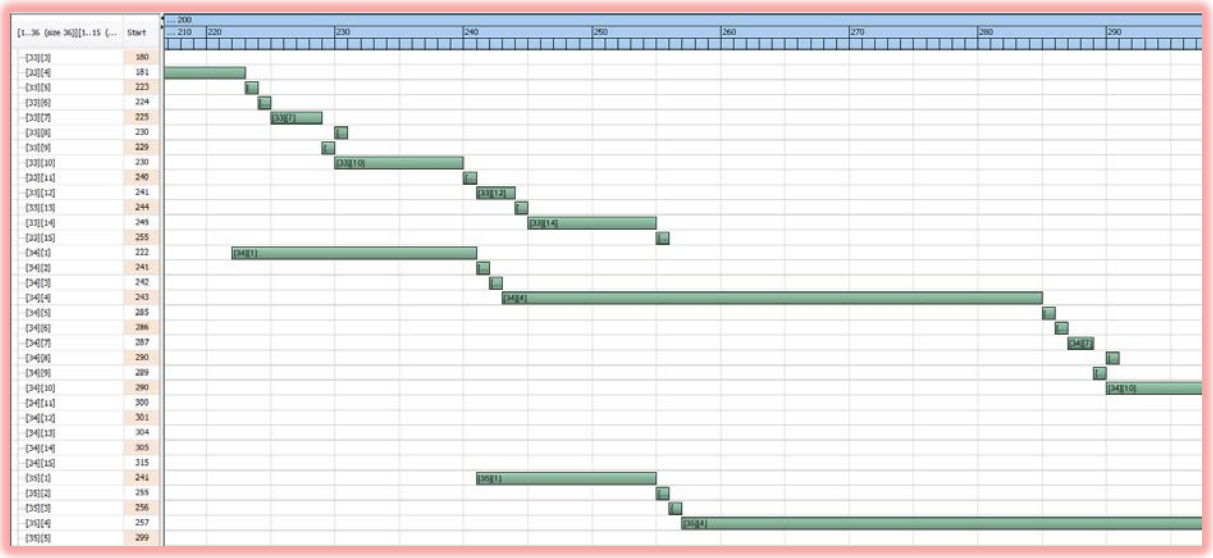


Figure 21. Optimized process plan Gantt chart constrained with available resources.

Conclusions

We have build out the support for META-X languages and iFAB languages. For META languages we focus on:

- CyPhy
- Modelica
- QML
- SysML
- CAD (specifically FreeCAD/OpenCascade and SolidWorks as an example)

For iFAB Languages we focus on

- MCPML (Boeing)
- M-SysML (GTech)
- FDL/PDL/MML (PennState)

To allow us to do cross-language analysis across these languages while still maintaining uniform semantics across models, our work included a layer of **Foundation** languages. To support Manufacturability Analysis we showed languages for **Products**, **Foundry Resources** as well as **Manufacturing Processes** and **Process plans**. We use these as foundation languages so that other language models can map to these languages for analysis.

These foundation languages are VHDL like languages and are built in such a way that other iFAB and META languages can provide content expressed in these foundation languages. In particular, these are some examples of content that can be expressed in the foundation languages:

  - **Product**: Product (BOM, assembly), Subassembly, Part, Seam Type (between parts), Seam.
  - **Manufacturing Process**: Operation Type (pre and post conditions), Resource Type, Substitution (features).
  - **Foundry**: Foundry, Resource, Consumable
  - **Process Plan**: Process Plan, Instruction (seams, actuals), Alternative (cost), Schedule (time), Sequence (parallel, arbitrary)

Appendix 1: Manufacturing Annotation User Model

Jeff Henrikson
Intentional Software
Revised Jan 24, 2012

This document describes a user model for FANG participants labeling manufacturing process and other metadata into CAD models with supporting software. This document is not prescriptive about particular metadata required for particular manufacturability queries under a particular iFab model. This document prescribes nothing about geometric elements or relationships beyond the notions of AP203. A variety of feature recognition and feature-based design techniques should be implementable using the Manufacturing Annotation User Model.

The constituents of the Manufacturing Annotation User Model are:

An off the shelf CAD editor

A metadata editor

Suitable extension points and implementations.

Any metadata editor meeting this specification and the specification of the integrated iFab toolchain should be considered acceptable input in a FANG challenge, including but not limited to CyPhy, CyDesign, Intentional CLAMP, and Dassault META toolchain.

This document will use the term "strong identity" to mean an identity assigned by an editor to a document element which:

Is saved to disk in the editor's native format

Can be programmatically obtained by the editor's native API

Is unchanged by edits of sufficiently different elements of the document

Though outside the context of editors, a well known example of strong identity is the use of globally unique interface identifier (IID) in the Component Obect Model (COM) standard. A nonexample of strong identity is STEP "Instance name" (e.g. #123 at the beginning of a line), since they are typically generated from consecutive integers and thus can change drastically from edit to edit.

This document is primarily concerned with the notion of strong identity supported by the editors of off the shelf CAD systems.

The manufacturing process and other metadata will be edited in the metadata editor, typically open at the same time as a CAD editor.

AP203/XML support

To support a given off the shelf CAD editor, one of the following is required:

A.1) (Adapt CAD editor's existing AP203 export filter)

A set of macros for decorating CAD element strong identity information into a format transportable by the CAD editor's AP203 export. Note that with STEP's approach of identity

stopping with notions such as "Instance name" (e.g. #123), the (temporary) decoration will not be compliant with STEP. For example, in SolidWorks name properties will pass through the AP203 export.

A batch reading filter for the AP203 export (P28 or P21 embedding) and recovering the strong identities from the decorations. The result should be recorded in a P28 (xml) embedding with an extra XML attribute avmcadid=. The decorations for the strong identities may be removed so as to be conformant with AP203/xml, with the addition of the avmcadid attribute.

Even if an AP203 editor is not naturally inclined to export an identity for each solid, surface, edge, and vertex of the CAD model, the decorating macro and reading script should make a best effort to compose a strong identity from other strong identities. For example, an edge could be identified by the identities of two adjacent surfaces.

A.2) A custom AP203/xml export from the CAD tree with avmcadid attribute.

A.3) A lazy (demand-driven instead of batch) DOM-like view of A.2.

A.4) A lazy (demand-driven instead of batch) DOM-like view of A.2 supporting writes to the CAD store.

(optional) To the extent that feature information is present in the CAD document, the decoration macros may present it as duplicate elements for AP203 to be recovered by the reading filter. Note that AP203 does not prescribe feature definitions but AP224 can be used as a starting point.

Metadata XML

The metadata editor must be able to export its annotations as xml. It should use an xml attribute avmcadref= to record strong identities from CAD. Thus the avmcadid and avmcadref pair will follow the IDREF pattern of XML Schema:
http://books.xmlschemata.org/relaxng/ch19-77159.html

By combining the metadata xml and the CAD AP203 xml, we will have an annotated product model. The consuming application can join the references with definition pieces however appropriate.

Interactive labeling

To support the interactive labeling of a CAD model, the metadata editor and CAD editors must supply extension points. In general, different CAD and metadata editors would be expected to implement different extension points, depending on respective functionality.

All CAD editors should provide:

B.1) A programmatic means of obtaining a strong identity during user interaction. In scenario A.1, this could be by a special macro that implements strong identity copying to Windows clipboard. In scenario A.3 or A.4 it could simply be use of the avmcadid attribute.

B.2) A means to visually identify an element with a given strong identity. In cases where the identified element is offscreen, it should be made visible. A selection in the CAD editor may be one reasonable notation for visually identifying an element. In cases where strong

identities had to be composed from other identities (e.g. identifying an edge from two surfaces), the composition must be parsed for interpretation.
All metadata editors should provide:
C.1) A way to record a reference to a strong identity in the CAD editor. In scenarios A.1 or A.2, this could be done with a paste from Windows clipboard. In scenarios A.3 and A.4 ability to consume the described data source is required.
C.2) A method to invoke B.2 from a selection of a cad reference.
Any windows clipboard data should be transmitted with an IDataObject format string of "AvmCadref", and use "Text" only as a fallback.